Amendments to the Claims

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Presently Amended) A computer system configured to:
   - A) provide at least one task queue having a top end and a bottom end and in which can be stored and from which can be retrieved task identifiers, which identify tasks to be performed; and
   - B) for each provided task queue, employ a separate execution thread associated therewith to:
     - i) select repeatedly a current access mode from one of a LIFO access mode and a FIFO access mode in accordance with a mode-selection criterion; and
     - ii) perform dynamically identified tasks by repeatedly:
       - a) popping a task identifier from one of the top end ~~and~~ or the bottom end of that task queue in order to access that task queue in a LIFO access mode or a FIFO access mode in accordance with the current access mode;
       - b) so performing the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and
       - c) pushing onto that task queue task identifiers that identify any tasks thus found.

2. (Previously Presented) A computer system as defined in claim 1 wherein pushing occurs at , the bottom end of each provided task queue, popping in accordance with the FIFO access mode occurs at the top end of each provided queue, and popping in accordance with the LIFO access mode occurs at the bottom end of each provided task queue.

3. (Previously Presented) A computer system as defined in claim 1 wherein queue accesses in each provided task queue are circular.

4. (Previously Presented) A computer system as defined in claim 1 wherein the computer system is configured to provide a plurality of the task queues.

5. (Previously Presented) A computer system as defined in claim 4 wherein each said dynamically identified task is a garbage-collection task for performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects, and thereby identifying the tasks of performing similar processing for those other objects.

6. (Original) A computer system as defined in claim 5 wherein the task identifiers are identifiers of the objects associated with tasks that the task identifiers identify.

7. (Original) A computer system as defined in claim 6 wherein the task identifiers are pointers to the objects associated with the tasks that the task identifiers identify.

8. (Original) A computer system as defined in claim 4 wherein, in at least some instances, an execution thread associated with a task queue that is empty:

A) pops a task identifier from a task queue other than the one with which it is associated;

B) so performs the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

C) pushes onto the task queue associated with it task identifiers that identify any tasks thus found.

9. (Original) A computer system as defined in claim 8 wherein each said dynamically identified task is the garbage-collection task of performing, for a

given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

10. (Presently Amended) A compiler/interpreter that, in response to signals representing instructions that define operations in which memory for data objects is allocated dynamically, generates signals representing instructions that implement a garbage collector that operates in garbage-collection cycles of which each includes an operation that includes:

A) providing at least one task queue having a top end and a bottom end and in which can be stored and from which can be retrieved task identifiers, which identify tasks to be performed; and

B) for each provided task queue, employing a separate execution thread associated therewith to:

i) select repeatedly a current access mode from one of a LIFO access mode and a FIFO access mode in accordance with a mode-selection criterion; and

ii) perform dynamically identified tasks by repeatedly:

a) popping a task identifier from one of the top end ~~and~~ or the bottom end of that task queue in order to access that task queue in a LIFO access mode or a FIFO access mode in accordance with the current access mode;

b) so performing the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

c) pushing onto that task queue task identifiers that identify any tasks thus found.

11. (Previously Presented) A compiler/interpreter as defined in claim 10 wherein each garbage-collection cycle includes an operation that provides a plurality of the task queues.

12. (Original) A compiler/interpreter as defined in claim 11 wherein, in at least some instances, an execution thread associated with a task queue that is empty:

A) pops a task identifier from a task queue other than the one with which it is associated;

B) so performs the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

C) pushes onto the task queue associated with it task identifiers that identify any tasks thus found.

13. (Original) A compiler/interpreter as defined in claim 10 wherein the task identifiers are identifiers of the objects associated with tasks that the task identifiers identify.

14. (Original) A compiler/interpreter as defined in claim 13 wherein the task identifiers are pointers to the objects associated with the tasks that the task identifiers identify.

15. (Presently Amended) For performing dynamically identified tasks, a method comprising employing a computer system to:

A) provide at least one task queue having a top end and a bottom end and in which can be stored and from which can be retrieved task identifiers, which identify tasks to be performed; and

B) for each provided task queue, employ a separate execution thread associated therewith to:

i) select repeatedly a current access mode from one of a LIFO access mode and a FIFO access mode in accordance with a mode-selection criterion; and

ii) perform dynamically identified tasks by repeatedly;

a) popping a task identifier from one of the top end ~~and~~ or the bottom end of that task queue in order to access that task

queue in a LIFO access mode or a FIFO access mode in accordance with the current access mode;

b) so performing the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

c) pushing onto that task queue task identifiers that identify any tasks thus found.

16. (Previously Presented) A method as defined in claim 15 wherein pushing occurs at the bottom end of each provided task queue, popping in accordance with the FIFO access mode occurs at the top end of each provided task queue and popping in accordance with the LIFO access mode occurs at the bottom end of each provided task queue.

17. (Previously Presented) A method as defined in claim 15 wherein queue accesses in each provided task queue are circular.

18. (Previously Presented) A method as defined in claim 15 wherein step (A) includes providing a plurality of the task queues.

19. (Original) A method as defined in claim 18 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

20. (Original) A method as defined in claim 19 wherein the task identifiers are identifiers of the objects associated with tasks that the task identifiers identify.

21. (Original) A method as defined in claim 20 wherein the task identifiers are pointers to the objects associated with the tasks that the task identifiers identify.

22. (Original) A method as defined in claim 18 wherein, in at least some instances, an execution thread associated with a task queue that is empty:

A) pops a task identifier from a task queue other than the one with which it is associated;

B) so performs the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

C) pushes onto the task queue associated with it task identifiers that identify any tasks thus found.

23. (Original) A method as defined in claim 22 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

24. (Presently Amended) A storage medium containing instructions readable by a computer system to cause the computer system to:

A) provide at least one task queue having a top end and a bottom end and in which can be stored and from which can be retrieved task identifiers, which identify tasks to be performed; and

B) for each provided task queue, employ a separate execution thread associated therewith to:

i) select repeatedly a current access mode from one of a LIFO access mode and a FIFO access mode in accordance with a mode-selection criterion; and

ii) perform dynamically identified tasks by repeatedly:

a) popping a task identifier from ~~from~~ one of the top end ~~and~~ or the bottom end of that task queue in order to access that task queue in a LIFO access mode or a FIFO access mode in accordance with the current access mode ;

b) so performing the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

c) pushing onto that task queue task identifiers that identify any tasks thus found.

25. (Previously Presented) A storage medium as defined in claim 24 wherein pushing occurs at the bottom end of each provided queue, popping in accordance with the FIFO access mode occurs at the top end of each provided queue, and popping in accordance with the LIFO access mode occurs at the bottom end of each provided queue.

26. (Previously Presented) A storage medium as defined in claim 24 wherein queue accesses in each provided task queue are circular.

27. (Previously Presented) A storage medium as defined in claim 24 wherein the instructions cause the computer system to provide a plurality of the task queues.

28. (Original) A storage medium as defined in claim 27 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

29. (Original) A storage medium as defined in claim 28 wherein the task identifiers are identifiers of the objects associated with tasks that the task identifiers identify.

30. (Original) A storage medium as defined in claim 29 wherein the task identifiers are pointers to the objects associated with the tasks that the task identifiers identify.

31. (Original) A storage medium as defined in claim 27 wherein, in at least some instances, an execution thread associated with a task queue that is empty

A) pops a task identifier from a task queue other than the one with which it is associated;

B) so performs the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

C) pushes onto the task queue associated with it task identifiers that identify any tasks thus found.

32. (Original) A storage medium as defined in claim 31 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

33. (Presently Amended) A signal representing a sequence of instructions that, when they are executed by computer system, cause the computer system to:

A) provide at least one task queue having a top end and a bottom end and in which can be stored and from which can be retrieved task identifiers, which identify tasks to be performed; and

B) for each provided task queue, employ a separate execution thread associated therewith to:

i) select repeatedly a current access mode from one of a LIFO access mode and a FIFO access mode in accordance with a mode-selection criterion; and

ii) perform dynamically identified tasks by repeatedly:

a) popping a task identifier from ~~from~~ one of the top end ~~and~~ or the bottom end of that task queue in order to access that task queue in a LIFO access mode or a FIFO access mode in accordance with the current access mode ;

b) so performing the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

c) pushing onto that task queue task identifiers that identify any tasks thus found.

34. (Previously Presented) A signal as defined in claim 33 wherein pushing occurs at the bottom end of each provided queue, popping in accordance with the FIFO access mode occurs at the top end of each provided queue, and popping in accordance with the LIFO access mode occurs at the bottom end of each provided queue.

35. (Previously Presented) A signal as defined in claim 33 wherein queue accesses to each provided task queue are circular.

36. (Previously Presented) A signal as defined in claim 33 wherein the instructions cause the computer system to provide a plurality of the task queues.

37. (Original) A signal as defined in claim 36 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

38. (Original) A signal as defined in claim 37 wherein the task identifiers are identifiers of the objects associated with tasks that the task identifiers identify.

39. (Original) A signal as defined in claim 38 wherein the task identifiers are pointers to the objects associated with the tasks that the task identifiers identify.

40. (Original) A signal as defined in claim 36 wherein, in at least some instances, an execution thread associated with a task queue that is empty:

A) pops a task identifier from a task queue other than the one with which it is associated;

B) so performs the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

C) pushes onto the task queue associated with it task identifiers that identify any tasks thus found.

41. (Original) A signal as defined in claim 40 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

42. (Presently Amended) A computer system comprising:

A) means for providing at least one task queue having a top end and a bottom end and in which can be stored and from which can be retrieved task identifiers, which identify tasks to be performed; and

B) for each provided task queue, means for employing a separate execution thread associated therewith to:

i) select repeatedly a current access mode from one of a LIFO access mode and a FIFO access mode in accordance with a mode-selection criterion; and

ii) perform dynamically identified tasks by repeatedly:

a) popping a task identifier from ~~from~~ one of the top end ~~and~~ or the bottom end of that task queue in order to access that task queue in a LIFO access mode or a FIFO access mode in accordance with the current access mode ;

b) so performing the task thereby identified as, in at least some instances, to find one or more further tasks to be performed; and

c) pushing onto that task queue task identifiers that identify any tasks thus found.

43. (Previously Presented) A computer system as defined in claim 1 wherein the mode-selection criterion is based on the number of entries in the task queue.

44. (Previously Presented) A compiler/interpreter as defined in claim 10 wherein the mode-selection criterion is based on the number of entries in the task queue.

45. (Previously Presented) A method as defined in claim 15 wherein the mode-selection criterion is based on the number of entries in the task queue.

46. (Previously Presented) A storage medium as defined in claim 24 wherein the mode-selection criterion is based on the number of entries in the task queue.

47. (Previously Presented) A signal as defined in claim 33 wherein the mode-selection criterion is based on the number of entries in the task queue.